CLAIMS

1. A method used in managing a serverless distributed file system, the method comprising:

managing directories of the file system using Byzantine groups; and

managing files within the directories without using Byzantine groups.

2. A method as recited in claim 1, further comprising managing files within the directories by saving replicas of the files to fewer computers than exist in the Byzantine groups.

3. A method as recited in claim 1, wherein each directory includes one or more directory entries corresponding to one or more of the files, and wherein each directory entry includes:

an identification of the file;

an identification of a plurality of computers where replicas of the file are stored; and

file verification data.

4. A method as recited in claim 3, wherein the file verification data comprises a hash value generated from applying a cryptographically secure hash function to the file.

5. A method as recited in claim 3, wherein the file verification data comprises a file identification number, a file version number, and a name of a user whose signature is on the file.

6. A method as recited in claim 1, wherein the directories are managed using a hierarchical namespace.

7. A method as recited in claim 1, wherein each of a plurality of computers in the serverless distributed file system need not trust the other ones of the plurality of computers.

8. A serverless distributed file system comprising:

a plurality of computers;

a first set of the plurality of computers operating to store directory information for the file system, wherein each computer of the first set is part of a directory Byzantine group; and

a second set of the plurality of computers operating to store replicas of the files in the file system, wherein for each file stored in the file system a plurality of replicas of the file are stored on the second set of computers, and wherein the quantity of replicas is less than the quantity of computers in the Byzantine group.

9. A serverless distributed file system as recited in claim 8, wherein one or more of the plurality of computers are in both the first set and the second set.

10. A serverless distributed file system as recited in claim 8, wherein the directory information includes a plurality of directory entries, and wherein each directory entry on a computer in the first set includes an indication of each computer in the second set where a copy of a file corresponding to the directory entry is located.

11. A serverless distributed file system as recited in claim 8, wherein the replicas comprise encrypted versions of the file being stored.

12. A method comprising:

generating a directory entry corresponding to a file to be stored in a serverless distributed file system;

saving the directory entry to each of a first plurality of computers that are part of a Byzantine-fault-tolerant group; and

saving the file to each of a second plurality of computers, wherein fewer computers are in the second plurality of computers than are in the first plurality of computers, and wherein at least one of the second plurality of computers is not part of the Byzantine-fault-tolerant group.

13. A method as recited in claim 12, further comprising:

receiving, prior to saving the file to each of the second plurality of computers, the file in encrypted form.

14. A method as recited in claim 12, further comprising:

receiving, prior to generating the directory entry, an encrypted file name of the file.

15. A method as recited in claim 12, wherein at least one of the second plurality of computers is part of the Byzantine-fault-tolerant group.

16. A computer comprising:

a processor;

a memory coupled to the processor; and

wherein the memory is to store a plurality of instructions to implement a file system using a hierarchical namespace to store files, wherein the file system is distributed across a plurality of computers including the computer, wherein each of the plurality of computers can operate as both a client computer and a server computer, wherein each of the plurality of computers need not trust the other ones of the plurality of computers, wherein files and corresponding directory entries are stored in the file system, and wherein for any given file fewer copies of the file are stored than are copies of the corresponding directory entry.

17. A computer as recited in claim 16, wherein the file system stores directory entries using Byzantine groups and objects corresponding to the directory entries without using Byzantine groups.

18. A computer as recited in claim 16, wherein the computer is part of a directory group responsible for managing a set of directories in the hierarchical namespace, and wherein the plurality of instructions are further to cause the computer to identify a new directory group and delegate responsibility for managing a subset of the set of directories to the new directory group.

19. A computer as recited in claim 18, wherein the plurality of instructions are further to cause the computer to:

identify a group of computers to be part of the new directory group;

generate a delegation certificate for the subset;

digitally sign the delegation certificate; and

issue the delegation certificate to the group of computers.

20. A computer as recited in claim 16, wherein the computer includes a cache of pathname to directory group mappings, and wherein the plurality of instructions are further to cause the computer to determine where to locate a file corresponding to a pathname in the file system by performing the following acts:

checking the cache to determine a mapping for a longest prefix of a desired pathname; and

if the entire pathname is mapped to a directory group using the cache, then accessing a member of the directory group to determine where to locate the file, and otherwise repeating the following until the entire pathname is mapped to a directory group,

obtaining, from a member of the directory group corresponding to the longest prefix of the desired pathname, mappings for a relevant subtree from the longest prefix, and

if the entire pathname is not mapped to a directory group using the relevant subtree from the longest prefix, then repeating the obtaining with the longest prefix being the previously used longest prefix concatenated with the relevant subtree.

21. A method comprising:

identifying a group of computers to which a subtree of a hierarchical namespace used to store files is to be delegated;

generating a delegation certificate for the subtree;

digitally signing the delegation certificate; and

issuing the delegation certificate to the group of computers.

22. A method as recited in claim 21, wherein digitally signing the delegation certificate comprises having the delegation certificate digitally signed by a plurality of computers.

23. A method as recited in claim 21, wherein the group of computer comprise a Byzantine-fault-tolerant group.

24. A method as recited in claim 21, wherein the delegation certificate comprises:

a first digitally signed certificate identifying another group of computers responsible for managing a namespace root of the subtree; and

a second digitally signed certificate allowing authorization of the group of computers to manage the subtree to be traced to the other group of computers responsible for managing the namespace root.

25. A method as recited in 24, wherein the second digitally signed certificate comprises:

an identification of a path below the beginning of another subtree previously delegated to a third group of computers, wherein the third group of computers are the directory group performing generating;

an identification of a root of the other subtree delegated to the third group of computers;

an identification of the subtree; and

an identification of the members of the group of computers.

26. A method as recited in claim 25, wherein the computers in the third group of computers are the same computers as in the other group of computers.

27. A method as recited in 24, wherein the first digitally signed certificate is digitally signed by a certification authority (CA).

28. A method as recited in 24, wherein the delegation certificate further comprises one or more additional digitally signed certificates allowing a certificate chain to be established from the second digitally signed certificate to the first digitally signed certificate.

29. A method implemented in a computing device of a serverless distributed file system, the method comprising:

checking a local cache of pathname to Byzantine-fault-tolerant directory group mappings to determine a mapping for a longest prefix of a desired pathname; and

if the entire pathname is mapped to a Byzantine-fault-tolerant directory group using the local cache, then accessing a member of the Byzantine-fault-tolerant directory group to determine where to locate a file corresponding to the pathname, and otherwise repeating the following until the entire pathname is mapped to a Byzantine-fault-tolerant directory group,

obtaining, from a member of the Byzantine-fault-tolerant directory group corresponding to the longest prefix of the desired pathname, mappings for a relevant subtree from the longest prefix, and

if the entire pathname is not mapped to a Byzantine-fault-tolerant directory group using the relevant subtree from the longest prefix, then repeating the obtaining with the longest prefix being the previously used longest prefix concatenated with the relevant subtree.

30. A method as recited in claim 29, wherein the mappings are obtained from the member of the directory group via a delegation certificate digitally signed by one or more members of the directory group.

31. A method as recited in claim 29, further comprising adding the mappings for the relevant subtree to the local cache.

32. A method as recited in claim 29, wherein the longest prefix includes one or more directories in addition to a namespace root.

33. A method implemented in a serverless distributed file system, the method comprising:

receiving a request to open an object with one or more selected locks;

checking whether the one or more selected locks conflict with a lock already granted to another application, wherein at least one of the selected locks represents the right to open the object without sharing an open mode; and

granting the request to open the object only if the one or more selected locks do not conflict with a lock already granted to another application.

34. A method as recited in claim 33, further comprising:

requesting that the device that holds the lock that conflicts with the one or more selected locks return the conflicting lock; and

granting the request to open the object if the conflicting lock is returned, otherwise denying the request to open the object.

35. A method as recited in claim 33, further comprising:

upgrading the request to a broader scope than indicated in the request;

checking whether the one or more selected locks of the broader scope conflict with a lock already granted to another application; and

granting the request to open the object with the broader scope only if the checking of the one or more selected locks of the broader scope indicate that no conflict exists.

36. A method as recited in claim 33, further comprising denying the request if checking whether the one or more selected locks indicate a desire to share the object for one or more operations that conflict with sharing indicated by another application that has previously been granted a lock or checking whether the one or more selected locks conflict with a lock already granted to another application indicate a conflict exists.

37. A method as recited in claim 33, wherein granting the request further comprises upgrading the request to a broader scope than indicated in the request and granting the broader scope.

38. A method as recited in claim 33, further comprising attempting to downgrade the previous lock to a narrower scope than previously granted to another application prior to denying the request if the one or more selected locks conflict with a lock already granted to another application.

39. A method as recited in claim 33, wherein the object comprises a file.

40. A method as recited in claim 33, wherein the object comprises a directory.

41. A method as recited in claim 33, wherein the one or more selected locks comprise an Open Read lock.

42. A method as recited in claim 33, wherein the one or more selected locks comprise an Open Write lock.

43. A method as recited in claim 33, wherein the one or more selected locks comprise an Open Delete lock.

44. A method as recited in claim 33, wherein the at least one selected lock comprises a Not Shared Read lock.

45. A method as recited in claim 33, wherein the at least one selected lock comprises a Not Shared Write lock.

46. A method as recited in claim 33, wherein the at least one selected lock comprises a Not Shared Delete lock.

47. One or more computer readable media having stored thereon a plurality of instructions to implement a serverless distributed file system, wherein the plurality of instructions, when executed by one or more processors, causes the one or more processors to perform acts comprising:

assigning responsibility for managing one or more directories to a directory group, wherein each member of the directory group is a computer participating in the serverless distributed file system; and

employing a plurality of locks to control access to objects in each directory, wherein the plurality of locks comprise,

a first set of locks to control opening of the objects, and

a second set of locks to control access to the data in the objects.

48. One or more computer readable media as recited in claim 47, wherein the second set of locks comprises:

a Read lock to control read access to the data in the objects; and

a Write lock to control write access to the data in the objects.

49. One or more computer readable media as recited in claim 47, wherein the first set of locks comprises:

an Open Read lock to control opening of the objects for reading;

an Open Write lock to control opening of the objects for writing; and

an Open Delete lock to control opening of the objects for deleting.

50. One or more computer readable media as recited in claim 47, wherein the first set of locks comprises:

a Not Shared Read Lock to indicate an unwillingness to share the ability to read the objects.

51. One or more computer readable media as recited in claim 47, wherein the first set of locks comprises:

a Not Shared Write Lock to indicate an unwillingness to share the ability to write to objects.

52. One or more computer readable media as recited in claim 47, wherein the first set of locks comprises:

a Not Shared Delete Lock to indicate an unwillingness to share the ability to delete the objects.

53. One or more computer readable media as recited in claim 47, wherein the plurality of locks further comprises an Insert lock to control creation of a new object with a particular name.

54. One or more computer readable media as recited in claim 47, wherein the plurality of locks further comprises an Exclusive lock to obtain all of the other ones of the plurality of locks for an object.

55. One or more computer readable media as recited in claim 47, wherein the objects comprise one or more files and one or more directories.

56. A serverless distributed file system comprising:

a plurality of computers;

a first set of the plurality of computers operating to store directory information for the file system, wherein each computer of the first set is part of a Byzantine-fault-tolerant group;

a second set of the plurality of computers operating to store replicas of the files in the file system, wherein for each file stored in the file system a plurality of replicas of the file are stored on the second set of computers, and wherein fewer computers are in the first set than in the second set;

wherein the first set of computers is configured to delegate management responsibility for a group of directories of the file system to a third set of the plurality of computers by,

generating a delegation certificate for the group of directories,

digitally signing the delegation certificate, and

issuing the delegation certificate to the third set of computers; and

wherein the third set of computers is configured to maintain management responsibility for the group of directories by employing a plurality of locks to control access to objects in each directory of the group, wherein the plurality of locks include,

a first set of locks to control opening of the objects, and

a second set of locks to control access to the data in the objects.